Naval Ocean Systems Center
San Diego, CA 92152-5000

Technical Document 1881
July 1990

Coordinates of a Life-Cycle Model for a Software System and Communicating the Need for Software Technology

NAVAL OCEAN SYSTEMS CENTER
San Diego, California 92152-5000

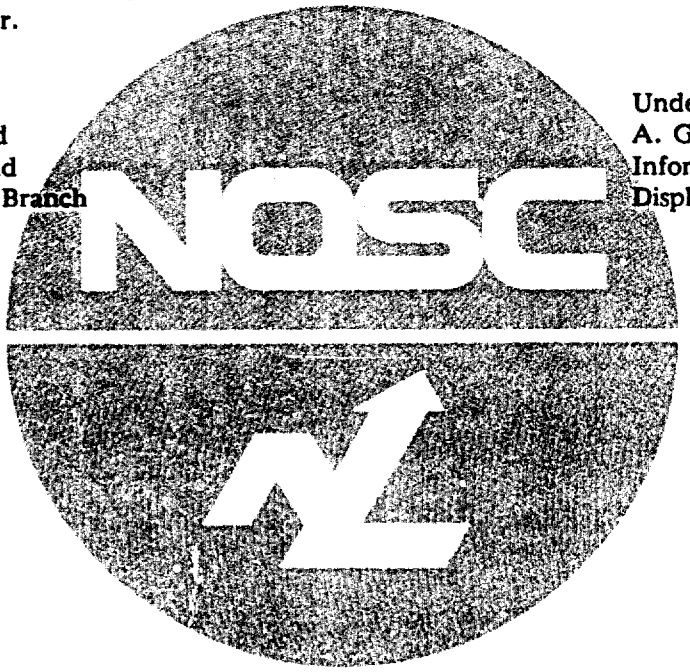
J. D. FONTANA, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

This work was performed for the Naval Ocean Systems Center, San Diego, CA 92152-5000, under program element 0602234N. Contract N66001-87-D-0179 was carried out by TECHPLAN Corporation, 5353 Mission Center Road, Suite 310, San Diego, CA 92108-0018, under the technical coordination of L. Sutton, Computer Systems and Software Technology Branch, Code 411, Naval Ocean Systems Center.

Released by
D. L. Hayward, Head
Computer Systems and
Software Technology Branch



Under authority of
A. G. Justice, Head
Information Processing and
Displaying Division

FS

ABSTRACT

On August 7-9, 1989, the Navy 6.2 Software Technology Project held a workshop meeting at the Naval Postgraduate School in Monterey, CA. Two parallel working groups were formed to address:

1) Evolving a consensus on the coordinates of a suitable software life-cycle model for the Navy;

2) Communicating the need for software technology.

The Working Meeting was opened by CDR Jane Van Fossen of the Office of Navy Technology who described the nature of software technology from a financial investment perspective. It was noted that software technology is a low profile budget item, with essentially no Navy 6.3A investment and a 6.2 budget that represents only about two million of the $430 million ONT budget. This level of investment in software technology seems inappropriate in light of the significance of software for major Navy systems. After CDR Van Fossen's remarks, the two working groups separated to pursue their respective topics.

This report summarizes the efforts of these working groups. Working Group 1 accomplishments are summarized in Section 2. Working Group 2 accomplishments are summarized in Section 3. Working group participants are listed in Appendices A and B respectively.

TABLE OF CONTENTS

SECTION 1.0

PROCEEDINGS OF WORKING GROUPS ONE AND TWO

COORDINATES OF A LIFE-CYCLE MODEL FOR A SOFTWARE SYSTEM

and

COMMUNICATING THE NEED FOR SOFTWARE TECHNOLOGY

1.1 Introduction.

On August 7-9, 1989, the Navy 6.2 Software Technology Project held a workshop meeting at the Naval Postgraduate School in Monterey, CA. Two parallel working groups were formed to address:

- Evolving a consensus on the coordinates of a suitable software life-cycle model for the Navy,
- Communicating the need for software technology.

This report summarizes the efforts of these working groups. Working Group 1 accomplishments are summarized in Section 2. Working Group 2 accomplishments are summarized in Section 3. Working group participants are listed in Appendices A and B respectively.

1.2 Initial Session - August 7, 1989.

The Working Meeting was opened by CDR Jane Van Fossen of the Office of Navy Technology who described the nature of software technology from a financial investment perspective. It was noted that software technology is a low profile budget item, with essentially no Navy 6.3A investment and a 6.2 budget that represents only about two million of the $430 million ONT budget. This level of investment in software technology seems inappropriate in light of the significance of software for major Navy systems. After CDR Van Fossen's remarks, the two working groups separated to pursue their respective topics.

SECTION 2.0

COORDINATES OF A LIFE-CYCLE MODEL FOR THE NAVY

2.1 Working Group 1 - Theme.

The Software Process Working Group 1 was convened to:

- Explore the possibility of a consensus among those whose work is centered in improving or in radically altering the life-cycle processes used in the development and evolution of large systems,
- Work out and agree upon the terms of that consensus,
- Consider how the evolution and emergence of that consensus can be supported and fostered both within the STP and within the Navy Laboratory community.

The convening of this working group is a project milestone.

2.2 Basic Working Group Directions.

Working Group 1 deliberations had two basic directions. The first dealt with the nature of the life-cycle model needed for Navy software and systems and its relationship to current life-cycle models and current research in process models. The second dealt with the features of life-cycle models and finding ways to adequately and thoroughly characterize such models. The group also considered several related issues.

2.2.1 Single vs Multiple Life-Cycle-Models.

The Working Group faced a number of questions concerning the nature and purpose of life-cycle models: Should there be a single model to which all will be required to conform? Should there be representative models which do not conform to a project's specific needs but which are sufficiently broad to demonstrate how many different features -- some of which do pertain to that project's needs -- can be incorporated into the same model?

These discussions resulted in the conclusion that there is a "CUT" or framework, or modeling framework, which can be defined and described both formally and otherwise. Such a cut may indeed be called a model (or even a Primary Life-Cycle Model), but is not in itself a sufficient definition of a completely determined application-specific model. Rather, it is the context for the necessary task of working out in detail the specific model needed for a given application. Moreover, there is not a single cut or framework, but several. Two such cuts were discussed at the Monterey meeting and briefly compared and contrasted. There are different approaches to the framework and they may yield different frameworks.

2.2.2 Elements of a Life-Cycle Model.

The Working Group faced questions concerning the various possibilities for identifying the elements which make up a fully defined model. These discussions resulted in the conclusion that within a given framework, the elements which together make up the model can be identified and defined in terms of invariants or constants which can be reused in various ways to create a range

of different models all within the given framework. These elements become, within that particular framework, a set of model construction primitives. The working group members engaged in a lively effort to identify some of the primitives and this effort resulted in certain incomplete but interesting sample lists of primitives. This effort concluded with a demonstration by example of how the primitives could be used to form a specific process model.

2.2.3 Related Issues.

There were several ongoing discussions of related issues and concerns. These were not intended to be central to the working group deliberations, but could not be completely excluded. These included technology, assessment of current process model approaches, and the DoD-STD-2167A standard.

Several working group members were keenly interested in the technology implied by or needed in support of the framework and the primitives which were being discussed. The technology discussions tended to support the need for one or more modeling languages and process modeling kits. One or two members held the opinion that it was possible to discover a set of primitives that hold for all possible frameworks, and that this set should be the basis for a formal modeling language and support system.

This optimistic assessment was not shared by all. Some felt that this judgment should be deferred until more information was available about the variety of frameworks. There was also a belief by some that the modeling support system would need to be some kind of a knowledge based system.

There was lively discussion about the waterfall model, its weaknesses and its strength, and its future. There was similar discussion about the spiral model. Finally, there was much discussion about how the Navy does software, about the perceived and the actual requirements of DoD-STD-2167A, and about the probability of the working group impacting this standard.

2.2.4 Process Research and Exploratory Development Consensus.

A consensus in software process research and exploratory development is not only possible but actually exists. It needs to be fostered, nurtured, and supported. One vehicle suited for this purpose is the continuation of the efforts of Working Group 1. There continues to be a diversity of opinion concerning how to do software process research. The most acceptable approach is for Working Group 1 to develop several specific models within a selected framework and then to develop a process modeling kit to support the selected framework. The Working Group 1 members agreed to reconvene in early FY 90.

2.3 Working Group 1 Findings.

This Navy 6.2 working group on Software Process Models seeks to understand how large and very large software systems are developed and used in complex organizational settings. As a community of interested researchers, this has led us to conduct a number of large software development experiments in laboratory settings. It has also led us to carefully review and analyze many published studies of software life-cycle models as well as software process frameworks. These studies, experiences, and analyses form the basis of our current position.

The Working Group 1 position that follows also takes into account the kinds of perplexing constraints that affect software developed for Navy applications. Such constraints include:

- o Use of Navy standard equipment.
- o Software programs embedded in larger systems which are often mission critical.
- o Software systems developed under contract to private firms.
- o Software development, use, and maintenance organizations are usually in different settings.
- o Systems are continually upgraded, repaired, and extended in response to changing requirements.
- o Current approaches to software developed for the Navy do not capture enough information of "how things get done" and thus may not be deployed, used, and supported as originally envisioned.

2.3.1 Need for a Framework of Software Construction Models.

In general terms, we hold that there should not be a single software life-cycle model to guide all possible Naval software system development efforts. The traditional software life-cycle model (the so called "waterfall model") and existing software development standards are crude indicators of software development processes. Further, they provide minimal guidance, lack rigorous formulation, and suggest little or no automation in the engineering ever more complex software systems.

We hold instead that what is needed is a framework for constructing models of software development processes. Such a framework (called elsewhere a Primary life-Cycle Model) serves as a context in which to address specific features of the application domain, the development team and its resources, budget and schedule limits, the application requirements, and the like, as well as the goals and objectives of its intended uses. Existing military software standards, such as DoD-STD-2167A, can be accommodated within such a framework. Further, such models should be formulated in terms that encourage more rigorous software process characterization and guidance, as well as computational formalization to enable systematic analysis, query, and reasoning. These are the characteristics of a new generation of software development models that can be more directly managed in ways that encourage more effective automation leading to increased software productivity and quality.

2.3.2 Understanding the Process of Large Software System Development.

We hold that the process of how large software systems are developed, used, and maintained over time (software evolution) must be understood in terms of products, processes, and settings. At present, we find classes of models for understanding software evolution in terms of software product development, software production processes, and software production settings. In general all of these models represent alternatives to the traditional software life-cycle model. Each of the classes of models has its strengths and weaknesses which can be described at length. However, our point here is to say that a contemporary model of software evolution should draw from each of the three classes of models and their interrelationships and from the traditional life-cycle model. In other words, we seek to model the evolutionary course of a particular software system in ways that indicate what gets produced, how, where, when, why, and by whom.

2.3.3 Navy Sponsorship of Software Process Research.

We note that Navy has helped to pioneer the sponsorship of applied research and exploratory development of such frameworks in projects at laboratories, at the Universities of California at Irvine and Southern California, and in industrial settings. Based upon these efforts together with the discussions of this working group, we can now present an initial outline of what issues must be addressed by a framework for modeling software processes.

2.4 Elements or Coordinates of the Software Life-Cycle.

The following subparagraphs provide a discussion of the discrete elements or coordinates of the software life-cycle in terms of software products, production processes, and production settings:

2.4.1 Software Products.

Software products represent the information-intensive artifacts that are incrementally constructed and iteratively revised through a software development effort. In this regard, the settings where software systems are produced and applied and the production processes that create them are themselves software products, i.e., products of prior or surrounding system development processes and settings. For example, the new skills that engineers can acquire through the development of a new software application are a product of a software development effort.

2.4.2 Production Processes.

Software processes represent networks of tasks that progressively transform available resources, software system requirements, and intermediate products into finished and assembled information products, e.g., life-cycle documents. Software evolution is not a single process. Instead, it involves many intersecting sub-processes which may be followed nondeterministically under local project circumstances. Therefore, we choose to view software production processes as a potentially non-linear sequence of tasks. These tasks can be decomposed into actions which in turn can be further decomposed into primitive actions. Such a decomposition is suggestive, not definite, except to say that software processes occur across many levels of activity, detail, and organizational administration. Thus, models of software production processes must be capable of abstract representation ranging from macroscopic activities to microscopic behaviors.

2.4.3 Production Settings.

The settings of software evolution can be modeled in terms of the people or programs ("execution agents") who perform production processes to produce the products. These agents can play single or multiple roles during a software development effort. Further, their role might be determined by their availability, participation in other organized roles, security access rights, or expertise. A role represents the set of skills (i.e., a reliable and robust operational plan) needed to perform some software production task. We often find, for example, software developers in the role(s) of "specification analyst," "coder," or "QA manager." Further, in order for an agent in a particular role to perform her/his task, a minimum set of resources and product/process requirements must be provided for task completion. Once again, the descriptions of which agents play which role in performing what tasks with what resources can also be modeled as interrelated attributed objects that can be created, composed, and managed in ways similar to those for software products and processes.

2.5 Process Network Decomposition.

Let us consider the suggested process network decomposition suggested above. Tasks correspond to development activities such as "develop functional specifications." Actions correspond to the units of the task such as "write the introduction section of the functional specification document." Then, primitive actions correspond to the units of an action that require invocation of a computer system or software tool command sequences such as "execute functional specification processor on <file.spec>." Following this approach to software process modeling, we see that such process descriptions can themselves be viewed as information entities with control structures that organize the flow of data (i.e., software objects) through a hierarchy of abstract activities. Further, these software process descriptions are software products which are created and refined by other processes possibly in other settings. Therefore, they can in a sense be constructed and managed as other software products.

2.6 Most Common and Tangible Software Products.

The most common and tangible software products are, however, software life-cycle documents. Requirements, specifications, software designs, source code, test cases, user manuals, and the like characterize what the developed software system is supposed to do, how it does it, how it was developed, how it was put together and validated, and how to install, use and maintain it. Rather than simply viewing these documents as text files, it is possible to view these documents as the composition of more basic software objects, such as individual requirements specifications, a module's detailed design, a single test case, etc. Given such a view, it is possible to construct, browse, and query classes of objects whose attributes can be interrelated or composed with other objects. In this way, we can recognize that all software objects can be modeled as information entities that include object type, structure, relations, and values. With the production setting in mind, it is possible when structuring software documents to identify and distribute sets of related objects being constructed, tested, or reviewed to teams of software developers working concurrently or in parallel.

2.7 Conclusions.

In conclusion, we reiterate our position: contemporary models of software evolution must account for the products, production processes, and settings, as well as their interrelationships that arise during software development, use, and maintenance. Such models can therefore utilize features of traditional software life-cycle models as well as those of automatable software process models. Finally, we have described an initial approach to the construct and support of models of software evolution through the use of software information structures that can be composed and managed within an advanced software engineering environment. While such a framework is yet to be fully articulated and exercised, its development, application, and transition to real development projects appear to be within reach in the next few years.

SECTION 3.0

COMMUNICATING THE NEED FOR SOFTWARE TECHNOLOGY

3.1 Working Group 2 - Theme.

The need for a workshop that deals with the problems of communicating the need for software technology research to decision makers who often have little or no software expertise.

In order to establish the theme for the working group, a set of introductory remarks were prepared and forwarded to as many of the expected participants as possible prior to the working meeting. These remarks served as a stepping-off point for the briefs and discussions of Working Group 2 and are included as Appendix C

3.2 First Working Group 2 Session - August 7, 1989.

The first Working Group 2 session took place on August 7, 1989. This session consisted primarily of presentations by various working group members. The following subparagraphs address these presentations and the discussions that followed.

3.2.1 Presentation by Thomas Conrad, NUSC.

In keeping with the announced plan, Working Group 2 began its deliberations with short presentations by several participants. The first of these was by Thomas Conrad of the Naval Underwater Systems Center (NUSC). This presentation focussed on problems experienced when attempting to communicate software technology issues to major program managers at the Navy System Commands. Among the highlights of the resulting discussion were:

a. the lack of a common vocabulary for dealing with software is a serious problem. Even the most well known innovators and leaders in the industry often disagree on basic software terminology.

b. The acquisition of huge, complex real-time software systems on firm fixed price contracts is a mixed blessing.

c. There is a tendency by many managers to seek to avoid dealing with difficult software issues until as late as possible in the development cycle with a result that many software products (especially documents) are treated as ancillary rather than critical path, and software qualities such as reliability tend to become hoped for results rather than built-in characteristics.

d. Frequent, well meant changes to requirements and design specifications wreak havoc with complex software system development schedules and product quality.

e. There is a real desire for a "silver bullet" to solve the software problem despite the unavailability of any cure-all and, in the absence of such a silver bullet, many managers are now asking what they should do to address the software problems they recognize. Risk Management is a topic of significance with respect to software.

3.2.2 Presentation by Jay Crawford of the NWC.

The second presentation was by Jay Crawford of the Naval Weapons Center. Several potential reasons for the lack of a perceived need for software technology within NAVAIR were given:

- o Not many new starts involving significant software are planned,
- o A tendency to tell program managers what they want to hear,
- o The over selling of software technology in the past,
- o Program managers are promoted for staying on schedule and within budget and not for taking risks on software technology improvements,
- o The tendency of program managers to come and go many times over the life-cycle of a software product.

It was suggested that the need for software technology could be better understood by developing data on the true cost of software (by tracking actual costs and maintaining a database for comparison), by providing more analogies to hardware, by improvements in program manager training, and by reducing the turnover rate for program managers.

3.2.3 Discussion on NWC Presentation.

In the ensuing discussion, several points were made.

a. It was noted that the Defense Systems Management College was doing better in providing training for program managers that includes software technology perspectives. On the other hand, the Navy continues to have difficulty filling its quota of training slots at DMSC.

b. There was a discussion of Ada shadow projects as vehicles for understanding the cost and benefits of software technology and, in particular, of Ada. It was concluded that shadow projects typically achieve implementation of only a small subset of a "real" system, and even that accomplishment is at significant expense. As a result, comparisons between the real program and the shadow program are difficult and conclusions are often suspect.

c. An opinion was voiced that there has been more success at transferring technology to other technologists than to Navy program managers.

d. Finally, there was a discussion of the nature of software technology. Does it encompass tools, processes, or both? It was mentioned that technology itself has in the past been viewed in terms of human modifications to either matter or energy, and that perhaps now it is more appropriately viewed in terms of modifications to information which itself lacks both mass and energy.

3.2.4 Presentation by Robert Wasilausky of the NOSC.

The third presentation was given by Robert Wasilausky of the Naval Ocean Systems Center (NOSC). He related that currently the software development process is receiving the most attention from program managers. One success story has been the establishment of a Software Engineering Process Office at NOSC. The Office spans the various departments at NOSC and is charged with the oversight of the soft ware process used by all software intensive programs and

the establishment of a centralized database of experiences with various software engineering processes. Several other points were made during the ensuing discussion:

a. There is a need to develop an effective process and to capitalize the tooling to support the process in industry.

b. There is a need to define the appropriate Navy laboratory role as regards Software Support Activities (SSA) and to consider whether an SSA is a mechanism for exposing software technology needs.

It was noted that perhaps the best technology transfer mechanism so far has been the software technologists themselves moving on to different jobs and taking their skills with them to new applications. It was also pointed out that we continually find ways to do the software technology job even without identifiable funds sufficient to the task. This has the deleterious effect of leaving the impression that in reality no significant funding of software technology is necessary.

3.2.5 Presentation by William Sweet of the SEI.

The final presentation of the first session was given by William Sweet of the Software Engineering Institute (SEI). His talk identified four comprehension gaps with respect to software technology:

- Between Researchers and Practitioners,
- Between Practitioners and Program Management Staff,
- Between Program Management Staff and Acquisition Professionals,
- Between Specialists and Generalists.

The need was cited for a common vision within DoD that includes a proper process, a defined product, a correct vocabulary, and an appropriate sensitivity to the important quality factors. Five approaches for bridging the comprehension gap were described. In increasing order of success experienced by the SEI, these are:

Guidebooks

Tutorials

Conferences/Seminars

Workshops

Workshop Reports

3.2.6 Example of Successful Workshop Report.

A relevant example of a successful Workshop Report was described, namely, the Proceedings of the Workshop on Executive Software Issues. The Workshop report addresses three categories of issues that should be brought to the attention of executives in the defense industry, government, and academia to propose resolutions to those issues. These categories include:

- o A national strategy for dealing with software issues,
- o Changes to acquisition policies and practices,
- o Issues associated with large complex systems.

The report was circulated at high levels and is judged to have made an impact.

3.2.7 General Discussion on Presentations.

In the discussions that followed, several Air Force initiatives aimed at improving software management were considered. These included the Bold Stroke project and a more recent effort implementing "software action teams." It was not clear whether the Air Force's "frontal assault" on the software management problem was bearing results proportional to the investment being made. It was clear, however, that the Air Force investment is far higher than the Navy's. In a related matter, it was pointed out that there is no mechanism for capital investment in software technology within the Navy Laboratory system. The Asset Capitalization Program at the NIF funded laboratories specifically precludes purchase of software assets with ACP funds.

At the conclusion of the presentations, the charter of the working group was restated as follows:

> "Explore some of the key dimensions of technology transition as understood by management, operational, technical, civilian, academic, and industrial personnel."

The group proceeded to elaborate a list of discussion questions appropriate to pursuit of the charter. The following discussion topics were selected:

- o What do we mean by Software Technology?
- o Are technological issues separable from management or organizational issues?
- o Which issues are appropriately addressed by Navy 6.2 funding?
- o Is the software technology vocabulary mature enough to permit clear transmission of ideas?
- o What have been the principal breakthroughs in software technology and how do they parallel hardware technology progress?
- o Can we cite good examples of success stories in selling the need for software technology?

3.3 Second Working Group 2 Session - August 8, 1989.

During the second session, the working group benefited from the presence of Frank Lamonica of the Rome Air Development Center. Mr. Lamonica had earlier presented a brief to all workshop participants concerning the Software life-Cycle Support Environment (SLCSE).

3.3.1 Description of SLCSE Development.

SLCSE is a software development environment framework which presents a consistent and common user interface accessing a set of software development tools supporting the full spectrum of DoD-

STD-2167A software life-cycle activities from initial requirements analysis to maintenance. The SLCSE was developed by RADC and represents a significant investment in software technology. Accordingly, there was much interest in the working group as to how such a major software technology effort had been successfully marketed to Air Force management.

3.3.2 SLCSE and Related Discussions.

In discussing the matter, it was learned that the Air Force differs significantly from the Navy in its approach to software technology. In the first place, the RADC laboratory has a recognized charter for software research and development. No Navy laboratory has a comparable charter. Secondly, a full 38 man division at RADC is annually funded out of a 6.2 Block Program to perform software technology work because of this charter. The head of this division has the flexibility to choose which technological thrusts to pursue under contract, and these proposals are briefed to a Deputy for Advanced Technology in the Pentagon who has decision authority.

The interesting element in the above is that the in-house laboratory people are continuously paid independent of success in marketing the proposals for contracted work. Thus they can spend years carefully planning a project without worrying where their salaries are coming from. This is in great contrast with the Navy laboratory mode of operation where almost no one is funded to do generic software technology work and researchers are largely forced to sell their skills annually to sponsors typically more oriented to development programs than to software technology per se. In the Navy, the key to success in marketing software technology proposals is obvious relevance to high priority Navy warfare areas. A possible conclusion is that growth of the Navy 6.2 software effort may require greater focus and less breadth in the future.

3.4 Answers to Questions Posed in Session 1.

At this point in the discussion, it was agreed to try to answer some of the questions posed during the first session.

3.4.1 Definition of Software Technology.

The first topic addressed was a definition of the term "software technology." After much discussion, it was decided that software technology embodies both process and product. Consequently, the following draft definition of software technology was devised:

- Principles and Guidelines for the development and support of cost-effective, high quality software.
- Software paradigms, algorithms, and products that extend the capabilities and performance of software intensive systems.

3.4.2 Identification of Software Technology Breakthroughs.

The next topic of discussion was the identification of the principal software technology breakthroughs to date. After some reflection, the following list of important developments was generated. It is neither complete nor chronological, but serves as a starting point for further work:

- High Order Programming Languages
- "Structured Programming"

- o Time-sharing
- o Virtual Memory Management
- o Object-Oriented Design
- o Information Hiding Principles
- o Cooperating Sequential Processes
- o Computer Communications Protocols
- o The E-R Model

3.4.3 Issues to be Addressed by the Navy 6.2 Software Program.

The next topic of discussion was identification of which issues are most appropriately addressed by a Navy 6.2 software program. It was stated that the aims of the 6.1 program and the requirements of the end users provide a natural context. In addition, the obvious need to avoid unnecessary duplication of efforts by others was reiterated. There was a perceived need to stimulate industry to make progress in areas of interest to the Navy. A specific need was cited to address capabilities that make the Navy a smarter buyer and owner of software. In addition, there is a requirement to evolve technologies relevant to very large, complex, real-time software systems, potentially distributed across heterogeneous processor sets, and that continuously evolve over long lifetimes.

It was concluded that technological issues considered in a vacuum yield unsatisfactory results. In many cases, management and organizational issues are strongly connected with technology issues and are appropriately addressed by a 6.2 software program. Indeed, software itself is largely an information management concept. Further, it was postulated that the different levels and types of management problems may require different treatment.

3.4.4 Discussion of Software Terms Needing Definition.

A short discussion of the software terms needing clear definitions yielded a very preliminary list that might be addressed in a glossary for managers. The list needs to be significantly expanded before any meaningful glossary could be produced. The preliminary list of important terms includes:

Environment	Toolset
Process	Method
Software Engineering	Case Tools
Software Maintenance	Post-Deployment Support
Software Evolution	Software Metrics
Runtime System/environment	Expert System
Workstation	Software Risk Management
Software Reusability	Software Productivity
Host/target Systems	Software life-Cycle

3.5 Third Working Group 2 Session - August 9, 1989.

The final session was given over to the development of a set of options for collecting and presenting information on software technology in ways that would help bridge the perceived communication gap.

3.5.1 Presenting Software Technology Information.

After considerable discussion the following possibilities were derived for collecting and presenting information on software technology in ways that would help bridge the perceived communication gap:

- o Elaborate a list of software technology issues and a set of important questions associated with each,
- o Generate a guidebook on the role of software technology in Navy systems - "The Layman's Guide to Software Technology,"
- o Vocabulary focus: produce a prose glossary of software technology terms,
- o Produce a description of software technology by analogy with hardware technology,
- o Develop a Software Technology Master Plan for the Navy that forecasts trends in requirements and trends in technology availability,
- o Use an example system as a vehicle for explaining software technology,
- o Produce an Executive level videotape as a marketing device to explain the Navy's 6.2 software program.

This collection of options was further refined by estimating the level of effort each was expected to require to be completed, a judgment of the order in which to produce each (assuming all were to be produced), and an estimate of the relative value of each in terms of contribution to resolving the perceived communication problem. The results of this effort are summarized in Table 3-1.

Table 3-1. Level-of-effort, order, and relative value of options.

OPTION	LOE REQUIRED	ORDER	VALUE
Issues With Questions	3 MM	2	1
Layman's Guidebook	12 MM	6	3
Prose Glossary	3 MM	3	2
Hardware Analogy	3 MM	1	2
Master Plan	9 MM	4	1
Definition By Example	6 MM	1	2
Executive Video	6 MM	5	3

3.6 Future Working Group 2 Meetings.

It was decided to continue the efforts of the working group for at least one more meeting to be held in the Washington DC area in the Autumn of 1989 so as to pursue one or more of the options discussed above.

APPENDIX A

Working Group 1 Attendees

Software Technology Project Working Meeting

1-3 August 1989

Name	Command/Org	Phone	E-Mail
Mark Moriconi	SI International Computer Science Lab Menlo Park, CA 94025	(415) 859-5364	moriconi@ csl.sri.com
Tom Cheatham	Software Options, Inc. 22 Hilliard St. Cambridge, MA 02138	(617) 497-5054	cheatham@ harvard.edu
Walt Scacchi	Univ. of Southern California Compuser Science Dept. Los Angeles, CA 90089-0782	(213) 743-7424	scacchi@ vaxa.isi.edu
Bob Westbrook	Naval Weapons Center Code 31C China Lake, CA 93555	(619) 939-5775 AV 437-5775	westbrook@ nwc.navy.mil
Michael Karr	Software Options, Inc. 22 Hilliard St. Cambridge, MA 02138	(617) 497-5054	mike%soi.uucp@ harvard.harvard.edu
Jim Oblinger	NUSC Code 2211,Bldg 1171-1 Newport, RI 02841-5047	(401) 841-3354 AV 948-3354	oblinger@ nusc-ada.arpa
Luqi	Naval PostGraduate School Monterey, CA	(408) 646-2735	luqi@ nps-cs.arpa
CDR Jane Van Fossen	ONT Code 227 800 N. Quincy St. Arlington, VA 22217-5000	(202) 696-4791	
Richard Jullig	Kestrel Institute 3260 Hillview Palo Alto, CA 94304	(415) 433-6871	jullig@ kestrel.kestrel.edu
David Wile	USC/Info. Sci. Inst. 4676 Marina del Rey Los Angeles, Ca 90292	(213) 822-1511	wile@ vaxa.isi.edu
Phil Hwang	NSWC White Oak Code U33 Silver Spring, MD 20903-5000	(202) 394-1351	hwang@ nswc-wo.arpa
Peter Feiler	Software Eng. Inst. Carnegie Mellon Univ Pittsburgh, PA 15213	(412) 268-7790	phf@ sei.cmu.edu

Name	Command/Org	Phone	E-Mail
W. Linwood Sutton	NOSC Code 411 San Diego, CA 92125	(619) 553-4082	sutton@ nosc-tecr.arpa
Leon Osterweil	UC-Irvine Information and Comp Sci Dept Irvine, CA 92717	(714) 856-4048	ljo@ ics.uci.edu
John Bilmanis	NSWC White Oak Code U042 Silver Spring, MD 20903-5000	(202) 394-1229	jbilman@ nswc-wo.arpa
Raymond Liu	NOSC Code 411 San Diego, CA 92125	(619) 553-4076	liu@ nosc.mil

APPENDIX B

Participants on Working Group 2

at the Navy 6.2 Software Workshop

William Sweet	Software Engineering Institute
Connie Heitmeyer	Naval Research Laboratory
Stan Wilson	Naval Research Laboratory
James Smith	Office of Naval Research
Robert Wasilausky	Naval Ocean Systems Center
Frank Lamonica	Rome Air Development Center
John Zahor	University of Washington
Jay Crawford	Naval Weapons Center
Thomas Conrad	Naval Underwater Systems Center
CDR Jane Van Fossen	Office of Naval Technology

APPENDIX C

The Initial Charge to Working Group 2:

Bridging the Software Technology Comprehension Gap

A dangerous thing has happened to computer software. We've become dependent upon it. Most of us are software addicts and don't even realize it. Software is running our cars; it manages our telecommunications; it entertains us with sophisticated video games; it handles our banking; it manages the restocking of grocery store shelves. It's everywhere.

But it's nowhere. We don't see the software computing our gas mileage in real time. We don't watch the software determine the routing of our telephone calls. When the video game fills our television screen with action filled graphics, the complex software that makes it happen is hidden away. When the automatic teller machine produces the cash we've requested, the software that verified our account number and debited our account was not only not seen at work, it was deliberately concealed from our attention. As we check out of the grocery store, we hardly notice the software that has recorded all our purchases, adjusted the inventory data, and generated orders for resupply.

The software is not obvious, it is not seen or heard, and it does not intrude on our consciousness. But it is there. What we see, what we hear, and what we feel is some piece of hardware: a display, a telephone, a control console, a sales register that prints a list of our purchases.

The Navy uses more software in its combat systems than any of the everyday applications mentioned here. The Navy's software is more complex. It has the highest demands on reliability. It has the longest lifetime. Navy combat systems depend on software.

But that software is still a mystery to many people involved with it. We should not be surprised. Software is the vital part you cannot see, hear, or feel. It is the thing that makes the system useful, but which has no physical manifestation as a system in and of itself.

As a consequence, most Navy program managers, acquisition officials, and budgetary decision makers have difficulty understanding the importance of software technology. Many have come to know that problems with software can result in significant cost and schedule impact. But since software as an entity is so poorly understood, software technology is all the more mysterious, focusing as it does on establishing better ways of producing an invisible product.

The objective of the working session will be to collect tutorial information on software technology as it relates to the Navy. The eventual publication of a layman's guide to software technology is a worthy objective, and it is hoped that the data collected here will provide a framework from which such a guide could evolve. In that regard, the organization of the information is nearly as important as the information itself, and it is expected that the working session will specifically address this issue. If the real question is how to explain the need for software technology, then several possible expository approaches come to mind.

One approach is to consider software technology in terms of Navy investment. In this case, the role of software technology is viewed in terms of 6.1, 6.2, 6.3 funding that addresses perceived software technology shortfalls. Alternatively, there is an organizational view that explains responsibilities of various Navy/DoD agencies for the availability of required software technology. Another approach is to structure the explanation of software technology according to a derived taxonomy of important issues (such as reliability, adaptability, production economy,

integrity, etc.) Still another view would be to describe the need for software technology on a life-cycle phase basis according to DoD-STD-2167A or some other process model for software generation. Whatever the view, or collection of views, selected, it seems necessary to address both trends in software requirements and trends in available technology.

At the August working session, it is planned that short (15 minute) briefs be presented by Navy air, surface, subsurface, and shore systems representatives. These talks would highlight experiences relating to difficulty in communicating the need for software technology. It is hoped to characterize common problems in this way to help focus the discussion. In addition, it is hoped that the Air Force perspective can be obtained from an RADC participant. Finally, a brief by the Software Engineering Institute on its experience with technology transition would round out the introductory briefs with a discussion on effective ways to communicate technology requirements.

After the introductory briefs, it is planned to pose the following questions for discussion:

- What are the top ten things about software that need to be understood?
- What are the principal reasons for Navy investment in software technology?
- What areas of software technology are developing at the slowest pace by comparison with Navy requirements?
- What are likely to be the new software technologies required in the 1995-2005 time frame?
- What form should explanatory material about software technology take?

The expectation is that the answers to these questions will form the core of the working group's output.

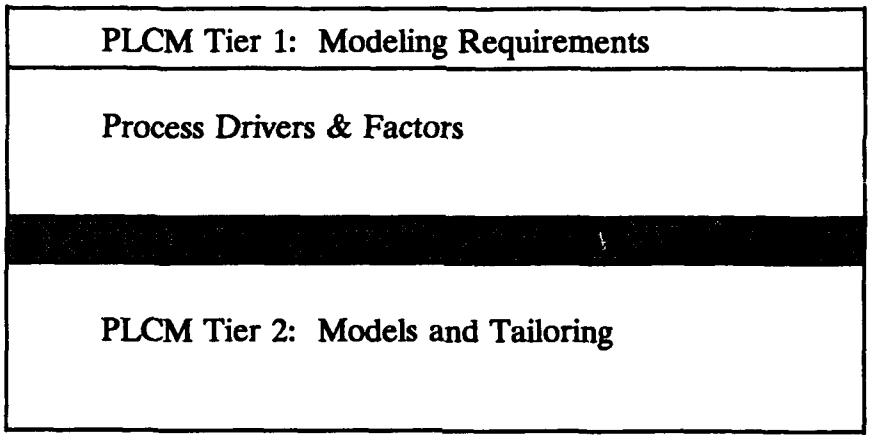
Letter Report
on the
Primary Life Cycle Model
(with Recommendation)

Software Engineering Technology Project (SETP) Project Management makes the following recommendation to N02D Block Management:

- That the process task under SETP be re-oriented to development of a Primary Life Cycle Model (PLCM) as described below (I)
- That the task be performed according to the approach described below (II)
- That the benefits of this recommendation by duly noted (III)

I. UNDERSTANDING THE PLCM

The PLCM embodies two principles and offers them as its key features, a modeling requirements principle and a modeling principle. These features are made available as a two-tiered model, each tier focusing on and featuring one of the principles.



Tier 1 Activity

The activity associated with Tier 1, as it relates to a particular project-specific application of the PLCM, consists of identifying the process drivers or factors which are important to the particular project, the relative importance of the factors, and how they are to be taken together (or not taken together).

Factors may include skill level and geography of human resources, schedule, cost, reliability in system performance, performance within time-constraints.

Reliability may be the key one project, while schedule may be more important for another.

Tier 2 Activity

For the specific project, this activity is concentrated on defining and describing appropriate models, consistent with and supportive of project-specific goals and requirements. The activity first maps the results of the tier 1 activity to tier 2 and then incorporates these mapped results into the model definitions and descriptions. Minimally, the descriptions cover definitions of processes to be used and their relationship to personnel, schedule and other resources.

This activity does not start from scratch. Tier 2 provides/assumes the availability of a number of models and modeling approaches consistent with DoD 2167. The models are in some sense parameterized (or at least "parameterizable"). The models are put together, and their parameters set according as determined by analysis of the mapped results from Tier 1 (ie. analysis of the factors).

II. APPROACH

A. Develop an initial PLCM and evolve it in an empirical and practical setting.

B. Provide a process modeling and tailoring computerized support system to facilitate the use of the PLCM approach.

C. Pursue the work in an orderly and coordinated fashion under the following activities:

- Initial PLCM Definition
- PLCM Evolution
- Research on Process Model Support Requirements
- Formal Definition of Process Support (PS) Systems
- Experimental Research with PS Systems and Technologies
- PLCM Transition

D. Involve the Navy Labs and Universities in the work process and also in the review of the results.

III. BENEFITS

Of the Model

- PLCM allows the process drivers to become keys in making decisions about a system development.

- PLCM is not limited to any particular model (e.g. Waterfall), but can accommodate and utilize a range of models and modeling approaches.

- PLCM is not a specific model but a modeling framework which guides and constrains the model definition effort, makes tailorability a more positively guided (and supported) activity, and shortens the time required to do model development and tailoring.

- PLCM provides a number of models which become 'Building Blocks' to be used in the model definition and development effort. This saves the model Definer-Builder the pain of having to start from scratch.

Of the Approach

- Reduces technical risks associated with the effort by its pairing with real Navy personnel and real projects, and by its use of researchers who focus on both the practical and the theoretical issues.

- Makes Success Comes in Stages

  1. The effort can succeed at Tier 1 level long before the problems associated with mapping of Tier 1 to Tier 2 are solved.

  2. The effort can succeed at bonding of Tier 1 and Tier 2 (ie. mapping of Tier 1 to Tier 2) and generation of the models as a labor-intensive manual activity.

  3. The effort can further succeed through providing computerized tools to support the activities identified above, which simplify the activities so that they can be performed by a larger class and persons, and which also automatically generate the models.

- Makes transition of result more probable through its tight coupling with the Navy labs.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1990	Final January 1990

4. TITLE AND SUBTITLE
COORDINATES OF A LIFE-CYCLE MODEL FOR A SOFTWARE SYSTEM AND COMMUNICATING THE NEED FOR SOFTWARE TECHNOLOGY

5. FUNDING NUMBERS
C: N66001-87-D-0179
PE: 0602234N
WU: DN088 690

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
TECHPLAN Corporation
5353 Mission Center Road, Suite 310
San Diego, CA 92108-0018

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Ocean Systems Center
San Diego, CA 92152-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NOSC TD 1881

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document presents the results of a software technology workshop meeting. Two parallel working groups were formed to address the following issues:

a. Evolving a consensus on the coordinates of a suitable software life-cycle model for the Navy.
b. Communicating the need for software technology.

14. SUBJECT TERMS
life-cycle models

15. NUMBER OF PAGES
30

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT